

A simulation study of simple local path planning and control for unmanned surface vehicles

Dimitrios Stergianelis
*School of Mechanical & Design
Engineering, University of Portsmouth
Portsmouth, UK*

Ya Huang*
*School of Mechanical & Design
Engineering, University of Portsmouth
Portsmouth, UK
ya.huang@port.ac.uk*

Matthew McConnell
*School of Mechanical & Design
Engineering, University of Portsmouth
Portsmouth, UK
matthew.mcconnell@port.ac.uk*

Hui Yu
*School of Creative Technologies
University of Portsmouth
Portsmouth, UK
hui.yu@port.ac.uk*

Ze Ji
*School of Engineering
Cardiff University
Cardiff, UK
jiz1@cardiff.ac.uk*

Abstract—The present study provides a fundamental but necessary review and implementation of collision avoidance and path planning techniques based on 2D plane kinematics of a small test vessel. The developed path planning logics first consider a single obstacle avoidance scenario, and then move on to tackle multiple obstacle avoidance based on more realistic obstacle and environment representations using multiple overlapping circles of various sizes and positions. The manuscript elaborates the development process step by step with all codes provided in order to introduce the path planning and simulation workflow as a laid tutorial for learners and researchers entering the field.

Keywords—unmanned surface vehicle, local path planning, simulation

I. INTRODUCTION

Building upon a previous study to develop a basic dynamic simulation framework to identify design parameters of unmanned surface vehicles (USV) in surge, sway and yaw axes [1], the present paper describes the implementation and comparison in the MATLAB-Simulink® environment of 1) a group of local path planning algorithms considering multiple waypoint and obstacle layouts; 2) hydrodynamic coefficients of vessel hull forms obtained from computational fluidic dynamic (CFD) simulations; 3) multiple waypoints and obstacles update. The study does not only provide with USV designers a starting point for their guidance navigation and control (GNC) system, but also some practical insights in selecting material to construct the surface vehicle as a mechanical system.

The role of the guidance component in the GNC system, [2], is to understand all input signals from the environment and derive an optimal route that is a) free from any obstacle or threat, b) the most ‘effective’ path, be it the minimum fuel consumption, time to reach, distance to reach, or the fastest speed. Path planning consists of three tasks.

The first is a map representation method to convert navigational map into geometric boundaries and constraints informing obstacles and potential waypoints. Given the varied performances and computational costs of different graph representation methods [3], to distinguish the areas available for navigation from the constrained areas, every obstacle can be represented as a circle with a fixed center position in a 2D map and a radius. This is the simplest approach to derive

obstacle avoidance. It reproduces the limitations presented in the allowable vessel positions.

Second, a global planning is usually represented using a graph method where the possible vessel locations (nodes) and the set of edges (paths) which the vessel can use to move from its current location to a new one. Dijkstra's graph and the A* algorithms and its heuristic extension produces solution to this scale, but all limited by the complexity of the obstacle domain, quality of the heuristic estimates and computational cost [5]. The present study will not seek global strategies, but concentrate on local planning that requires vessel manoeuvre up to 30 minutes to contact.

Third, the local path planning is introduced by the line of sight (LOS) technique to follow a sequence of waypoints known to the vessel and to comply with the International Regulations for Preventing Collisions at Sea (COLREGS, 1972) [5]. Artificial potential field (APF) method creates a repulsive potential field with magnitude inversely proportional to distance from the obstacle and an attractive potential field with magnitude proportional to distance from the target. The APF has potential non-convergency and local minima. The present work focuses on the LOS approach examining the obstacles relative to the path between waypoints. The solutions are developed and demonstrated for single obstacle and for multiple obstacle scenarios.

Strategies for single obstacle and multiple obstacle avoidance and path planning are treated separately with each having two methods based on LOS approach. The study has described the detailed reasoning and code for each case to allow designers of USV to adapt the planning tools to their specific needs of environment. The hydrodynamic simulation was described briefly as an example to guide the mechanical hull design process and setting the simulation parameters for controllers of the vehicle.

II. METHODOLOGY

The local path planning problem is defined by obstacles dictated by the operational environment and waypoints created by the planning algorithm. The obstacle domain of the USV is considered static in each planning algorithm with waypoints set to optimise each solution of path for simplicity [3, 4]. Once an efficient static planning algorithm is identified, with dynamically moving scene, any temporal update of obstacles can be executed ‘live’. The work considers a single obstacle first, and then moves on to address multiple obstacles.

* Correspondence author: Ya Huang (ya.huang@port.ac.uk)

A CFD simulation workflow does not only allow hydrodynamic coefficients to be extracted for different hull designs, and also enables the designers to use the fluidic parameters to implement simulations for selecting simple and effective control strategies [1].

All path calculation was performed using MATLAB® 2017b, and all codes are provided in the corresponding Annexes on the GitHub repository open to all public (<https://github.com/huangya17/Oceans-2020-boat-path>).

A. Dynamics simulation using CFD drag coefficients

Hydrodynamic drag forces were obtained in the surge, sway and yaw axes in CFD simulation. The simulations are setup according to the International Towing Tank Conference (ITTC) recommended procedures and guidelines for ship CFD [6]. The computational mesh consists of approximately 3.2 million trimmed mesh cells for grid independence. The volume of fluid (VOF) solver with a flat wave model is used to capture the water surface and wave damping, preventing wave reflection. The $k-\epsilon$ turbulence model was used to model the turbulence, and an implicit unsteady solver with a time step of 0.01 s was employed to extract the main flow features.

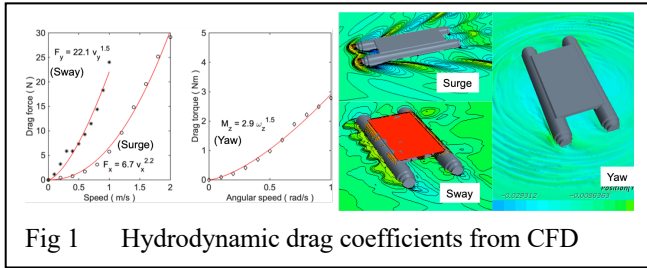


Fig 1 Hydrodynamic drag coefficients from CFD

B. Map and vehicle representation

A circumscribed circle with a diameter equal to the longest dimension of a single convex obstacle is used to represent single obstacle. For multiple obstacles, multiple circles producing minimal area but some overlap represent the obstacles. While there is no limit on the number and radius of the circles, these parameters should be considered with the task in question and computational effort.

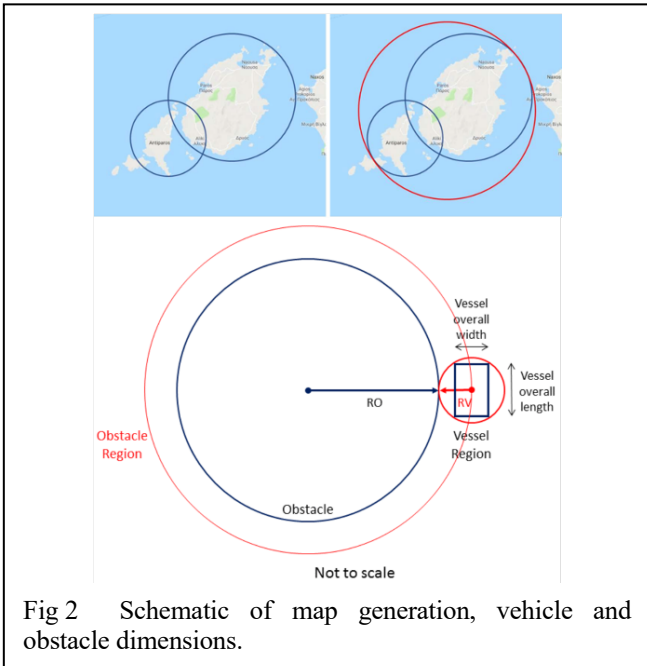


Fig 2 Schematic of map generation, vehicle and obstacle dimensions.

By applying a 2D plane dynamic model of the vehicle, surge (fore-and-aft translation in the x-axis), sway (lateral translation in the y-axis) and yaw (rotation about the x-axis) are modelled. The vehicle considered in this context [1] has a rectangular shape with a length of 1 m and an width 0.55 m. The diagonal of this rectangle is equal to 1.141 m representing the diameter of the vehicle region. The Vehicle Radius (RV) is then 0.571 m. These were the geometric parameters used to conduct the CFD simulation. A safety factor radius (RB) was assigned in addition to the original obstacle radius (RO). The determinant radius (RD) describing the entire obstacle region equates to the sum of RO and RB, where RB is represented by RV. RD is the non-navigable region marked by the red circle in Fig 2.

C. Projection method for a single obstacle

The projection method follows the start S (XS, YS) and through A and D to reach target T (XT, YT) in Fig 3a – the intersection points of the projection lines are manoeuvre

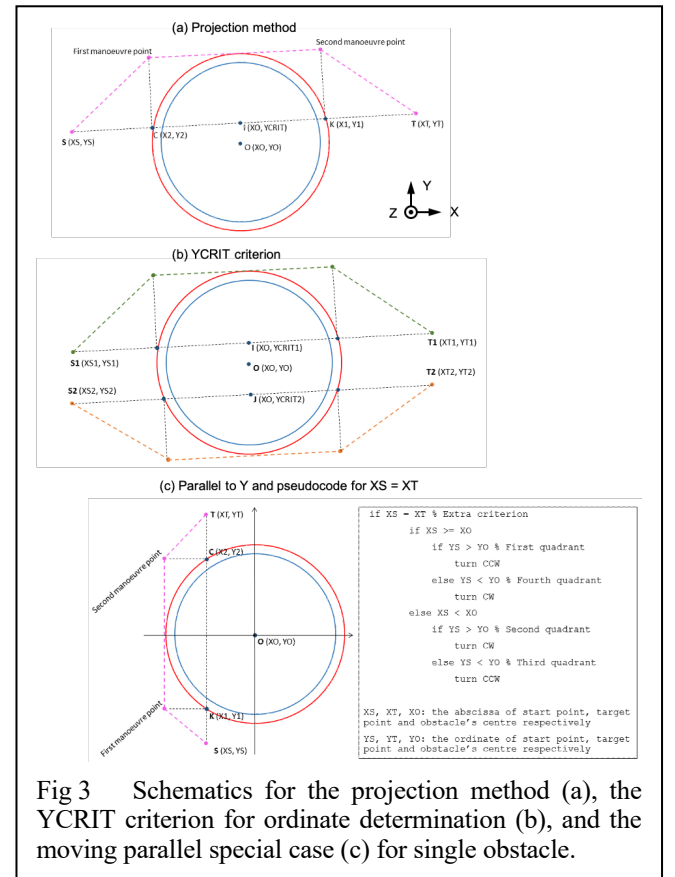


Fig 3 Schematics for the projection method (a), the YCRIT criterion for ordinate determination (b), and the moving parallel special case (c) for single obstacle.

points where the vehicle turns. The evaluation of the circle position was based on the YCRIT value: the ordinate of the straight line ST at abscissa equal to abscissa of point O.

At first, the intersection points C and K between the S and T straight line, and the characteristic red obstacle circle were determined. The projections of these points were used to define the manoeuvre points around the obstacle and to determine the tangential to the circle segment. The method makes use of the projections of the intersection points as manoeuvres points (Fig 3a). The MATLAB code was provided in Annex B on GitHub.

The distance between the centre of the circle and the straight LOS S1T1 or S2T2 is used to determine the shortest path. This evaluation is based on the YCRIT value, i.e. the

ordinate of the straight line at the abscissa of XO (Fig 3b). The values of YCRIT1 and YCRIT2 are calculated for the value of XO and the coordinates of I and J are derived. For the S1T1 path, the ordinate of I i.e. YCRIT1 is larger than YO, so the optimal path is to turn counter-clockwise (CCW) to follow the shortest route. For the S2 T2 path, the ordinate of J i.e. YCRIT2 is smaller than YO, so the optimal path is to turn clockwise (CW) for the shortest route.

In a special case when the vehicle moves parallel to the Y-axis (Figure 3c), the normal YCRIT criterion is no longer applicable as the YCRIT of the straight line does not have a fixed value and changes between YS and YT. A modified criterion is required. A series of checks between (XS, YS) and (XO, YO) respectively can determine the direction of turn for the shortest route. The pseudocode is provided in Figure 3c where the quadrants are determined by a coordinate system with its origin at the centre of the obstacle.

D. Tangent method for a single obstacle

In tight obstacle space the tangent method uses straight lines tangential to the obstacle. The tangent algorithm is developed after trying to apply the projection algorithm to tackle multiple-obstacle problems. The same YCRIT criterion is used to determine the shortest route around the obstacle and the direction of turn. The tangential lines from S to T were then introduced to the circle with radius RD in red (Fig 4a, b). The manoeuvre points are the intersection points between the three tangential lines (with MATLAB code in Annex C on GitHub).

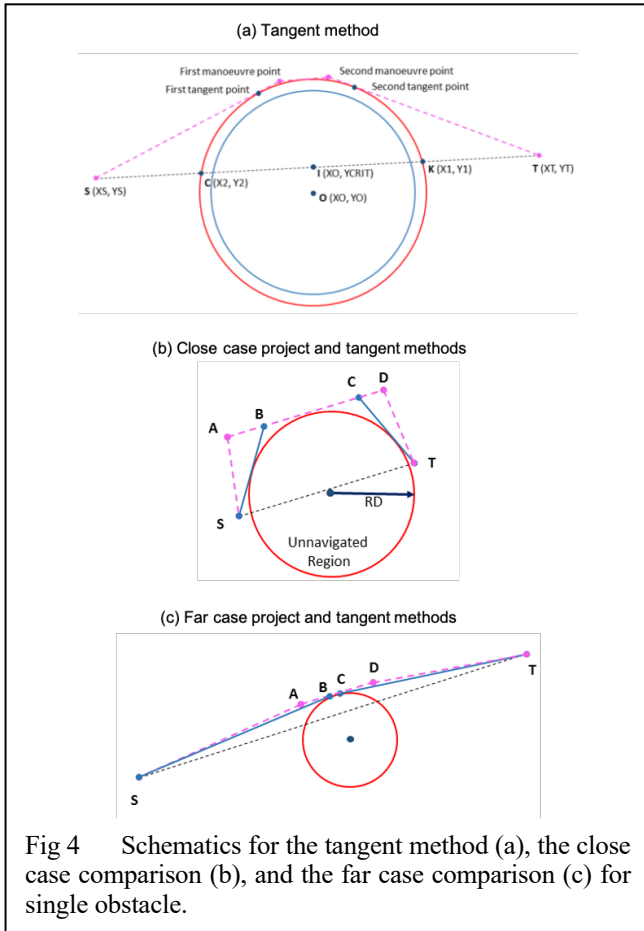


Fig 4 Schematics for the tangent method (a), the close case comparison (b), and the far case comparison (c) for single obstacle.

The main advantages of the tangent method – shorter route and manoeuvre points being closer to the obstacle, became less prominent when the start and target positions of the

vehicle are very close or very far away (Fig 4b, c). Both methods can clear the obstacle, but there is no clear advantage.

For multiple-obstacle problems introduced in the next section, the projection method is used as the basis mainly due to its simpler form. The geometrical complexity of the tangent method resulted in a more complicated code. With tightly placed obstacles, it is ‘safer’ for a basic controller to follow a simpler route with adequate margins.

Given the start and target positions of the vehicle and the non-navigable regions, the objective for multiple-obstacle avoidance is to design a path around this region with the shortest distance and minimal computational cost. Two different methods based on the projection method are outlined in the following sections.

E. Parallel method for multiple obstacles

The parallel method keeps the vehicle moving in parallel to the LOS line ST while avoiding non-navigable regions. In Fig 5a. The first leg S1 diverts the vehicle from the straight original course ST, by an angle f , to avoid the restricted region. The second leg S2 is tangential to the obstacle circle, also parallel to ST.

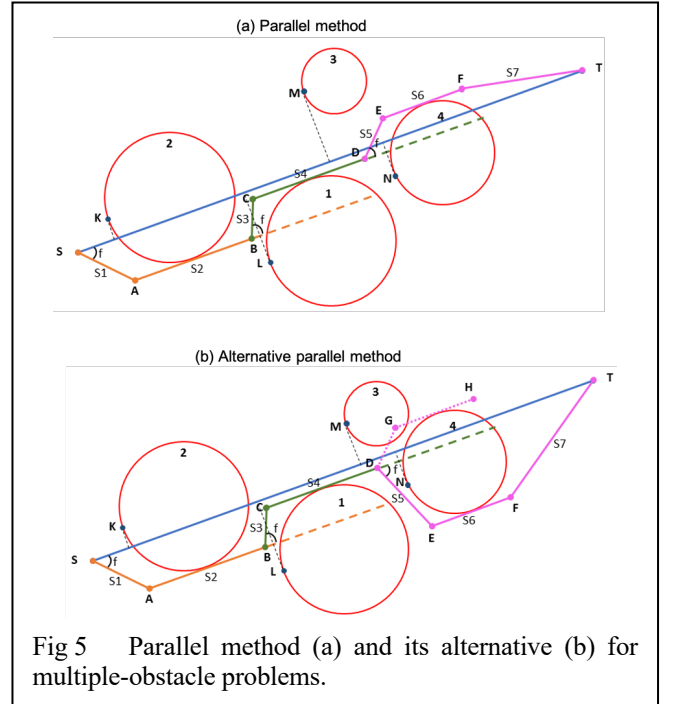


Fig 5 Parallel method (a) and its alternative (b) for multiple-obstacle problems.

In Fig 5a, point B marks the end of the first avoidance path. When the vehicle avoids obstacle number 2 (the closest to S), the algorithm for the parallel leg S2 decides if there will be an intersection with the next obstacle. Each time when an obstacle is avoided, this parallel leg is updated and used to determine the next avoidance manoeuvre. From the second obstacle onwards, if any obstacle intersecting with the parallel extension, the number of avoidance paths will be equal to the number of the obstacles discount the first obstacle. Each of these avoidance paths will have two segments. A final path with a single segment FT connects the last point F of the end of the last avoidance path with the target point T is formed. Table I provides an overview of the avoidance paths.

TABLE I. AVOIDANCE PATHS USING PARALLEL METHOD

Obstacle number	Avoidance paths	Segments
2	SAB (orange)	S1, S2
1	BCD (green)	S3, S4
3	No action	No action
4	DEF (magenta)	S5, S6

The pseudocode based on Fig 5a with a synopsis of the crucial points consists of four main steps and is presented below (with MATLAB code in Annex D on GitHub):

1) Determine the straight-line equation ($Y = a \cdot X + b$) connecting S and T positions.

2) Create an array having the obstacles rearranged according to their position with respect to the start point. Because the vessel is traveling parallel to the reference line, the minimum X value for every obstacle region should be calculated based on a coordinate system having its X axis parallel to the reference line. To achieve this:

- Calculate the XFI which is the abscissa of K, L, M, N in the global coordinate system.

- Calculate the XFR which the abscissa of K, L, M, N in the rotated coordinate system, where the X-axis is parallel to the reference line ($Y = a \cdot X + b$).

- Using the XFR as criterion the obstacles can be arranged according to their position for a vessel moving parallel to the reference line. A matrix with the order of the obstacles is created and is used in the loop execution.

3) Avoiding an obstacle region by creating a path consisting from two line-segments. Working on the n circle (a random circle):

Solve the ($Y = a \cdot X + b$) with the first (in the new rearranged matrix) circle. Use the Projection Algorithm to determine the turn angle $f(n)$.

Use the Projection Algorithm to determine the two manoeuvres points.

Update the position based on the above results; move on two line-segments, with the first one connecting the initial position with first manoeuvre point and the second one connecting the first manoeuvre point with the second manoeuvre point.

Update your start position (XS, YS) and the value of b (in the straight-line equation it is the Y-intercept) to get ready for the next loop iteration. These XS and YS are the coordinates of the end points of each avoiding path and will be the start points for the next path. The coefficient b is initially calculated based on the reference line from S to T, but in the end of each loop iteration it is updated to become the b on the tangential to the circle line (S2 from the first loop iteration).

4) A loop has to be executed as many times as the total number of the obstacles forming the domain.

In Fig 5b, obstacle 3 is closer to the other obstacles, making the initial path in Fig 5a invalid. Alternative algorithms are required.

Firstly, the heading of the vehicle needs to be checked against the next obstacle position. If the tangential extension line does not intersect the incoming obstacle circle, no action is required, i.e. between the S4 segment and obstacle 3). The algorithm has to execute the loop again to check the next obstacle, i.e. between the S4 segment and obstacle 4). This check can be performed by solving the system of straight line and circle equations.

Secondly, each obstacle avoidance path has to be checked against 'conflicts' from adjacent obstacles. When working on the n^{th} obstacle, no part of any avoidance path can intersect with $n-1^{\text{th}}$ or $n+1^{\text{th}}$ obstacle. If this check fails, the manoeuvre had to be performed using the alternative logic and not the projection method. The vehicle needs to turn in the opposite way from what the YCRIT commences, i.e. taking the longer route around the obstacle. In Fig 5b this problem is apparent with obstacle 4 and, and the alternative route is shown. Executing the avoidance algorithm in the loop for obstacle 4 results in the avoidance path DGH crossing the obstacle 3. As a result, the check failed, and the alternative DEF path is introduced to take a longer route around the obstacles.

If none of the alternative route is feasible, approximation techniques is required to group several close-by obstacles together or to guide the vehicle away from the reference line ST to clear the path to T. At this point, a global path planing technique is required, such as an A* algorithm. These techniques deviate the vehicle from its original approach and there was a clear need for a method able to handle the intersection between each segment and obstacle region.

F. Segment method for multiple obstacles

In the segment method, the algorithm constantly connects the current and target positions with a straight-line path, and the it focusses on avoiding the obstacles intersecting this path. Each time an obstacle blocking the straight-line path was avoided, using the projection method, three line-segments were created. In the next iteration each of the segments were checked to see if they intersected any obstacle and, if so, a new avoidance path was created for the specific segment and so on. Each obstacle that intersects with the straight line and is avoided increases the number of the line segments by two.

The pseudocode based on this algorithm consists of five main steps considering N obstacles with some blocking the desired original paths (with MATLAB code in Annex E):

1) Define the total number of line segments connecting the start S and target T positions, $K=1$.

2) Define the current line segment number, $k=1$.

3) Define the current obstacle number n, $n=1$ and the total number of obstacles, N.

4) While $k \leq K$ (for every line segment connecting S and T)

Check if the n^{th} obstacle (indifferent of its spatial location) intersects with the line segment connecting S_k and T_k (the start S_k and the end T_k points of the k^{th} line segment).

If there is no intersection:

If $n = N$:

That's the path, set $k=k+1$ to check the next line segment.

Check an d finish when $k=K$.

If $n < N$:

Set $n = n+1$ until $n=N$.

If there is intersection:

Use the projection method - this is the only obstacle.

Set $K=K+2$

- The total number of line segments has been increased due to manoeuvres needed

- For all line segments, check that none intersects with any obstacle. If there is, do more manoeuvres.

- For each line segment, S_k to T_k is considered in step 4).

Return to step 4) with a new K

5) Loop to check all obstacles.

An overview of the segment method is demonstrated using an example using four obstacles. The obstacle number, according to the obstacle array, is indifferent to obstacle positions and no sorting is required. The vehicle starts from S and heads to T in a straight line. This sets $K=1$ and $k=1$. Then all the obstacles were checked for intersections with the straight line ST , following the default order that they appear in the obstacle array. If no intersection is found between ST and any obstacle, the solution is found using $K=1$ and the desired path is the segment $S1$ shown in Fig 6a.

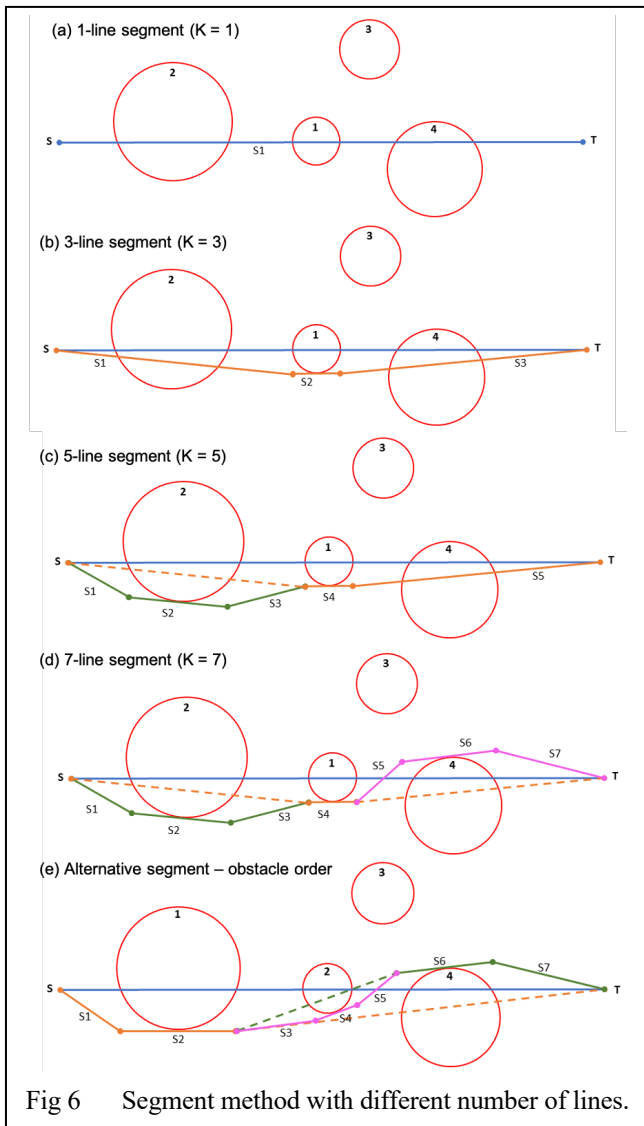


Fig 6 Segment method with different number of lines.

In Fig 6a, there is apparently intersections. According to the pseudocode $K = 1+2 = 3$ and a new route based on obstacle 1 is required as this is the first in the obstacle array. Avoiding obstacle 1 using the projection method, the orange path $S1-S2-S3$ is produced in Figure 6b. All segments need to be checked to see if they intersect with any obstacles, starting from $S1$ as it has the smaller value of $k=1$ and obstacle 1 as it has the smaller number of n . As $S1$ does not intersect with obstacle 1, the code continues to check if it intersects with obstacle 2. It does, so $K = 3+2 = 5$ and the projection method is used again to redesign $S1$ in order for this segment to clear obstacle 2. Fig 6c shows the result of this step with the green path consisting of three segments.

For $K=5$, five segments $S1, S2, S3, S4$, and $S5$ are created. Starting from step 3) of the pseudocode, the following checks are performed. Check one by one if $S1$ to $S5$ intersects with obstacle $n=1:N$, and increase n by 1 after each no-intersection check. If $S5$ intersects with obstacle $n=4$, set $K = 5+2 = 7$, and an avoiding manoeuvre is required. The result of the above iteration is shown in Fig 6d in magenta.

For $K=7$, seven segments $S1, S2, S3, S4, S5, S6$, and $S7$ are created. Starting step 3) of the pseudocode and making all the necessary checks, no intersection is found between the segments and the obstacles. The desired path is produced.

The driving force of the segment method is the iterative checks between generated segments and any obstacles in order to identify singularities. It examines each and every obstacle 'explicitly' by splitting the intersecting line segments into smaller sections so as to re-route a path. Having two obstacles close-by or overlay each other usually results in the avoidance path being designed based on the first obstacle intersecting the second obstacle due to the short distance between them. This obstacle configuration calls for an extra check and avoidance manoeuvre if the check fails.

The order of obstacles appeared in the obstacle array affects the final path produced. This is because the algorithm follows the order of the obstacles in the array. In the example presented in Fig 6a-d, by keeping the size and location of each obstacle and changing the order of the first two in the obstacle array, the alternative path can be produced in Fig 6e. The segment method examines all permutations of the obstacle array to ensure that all alternative paths are produced. If an obstacle does not intersect with any line segments, the algorithm skips the obstacle with no alternation. For each candidate path generated, the total trajectory length is calculated and compared. The shortest path is selected.

III. RESULTS AND ANALYSIS

This section compares the presented methods and their efficiencies based on the length of their routes, the positions of the manoeuvre points and the computation time. Vessel dynamics was not taken into account when defining the path. All results use a safety radius 0.571 m, the vessel region (RV). A dotted black line represents the start and target positions ST as a reference LOS line.

A. Representing the environment

All obstacle 'components' are modelled as circles; however, the user needs to decide how to represent the real environment with these simplistic circular models. The following three representation techniques can be applied.

1) One by one obstacle representation

Each obstacle is modeled with a circle and the coordinates of its centre combined with its radius (Fig 7). The blue circle RO represents the obstacle itself, and the red circle RD represents the obstacle region which is used in all calculations. Overlapping of the obstacle circles allows more flexibility for concave assembly of obstacles.

2) Merging multiple obstacles

Multiple obstacles forming a convex envelop can be combined into one larger circular obstacle. The benefit includes reduced computational effort. Whether this simplification can reduce the length of the route depends on the planning algorithm. In Table II, this scenario was simulated using five small obstacles. The segment method

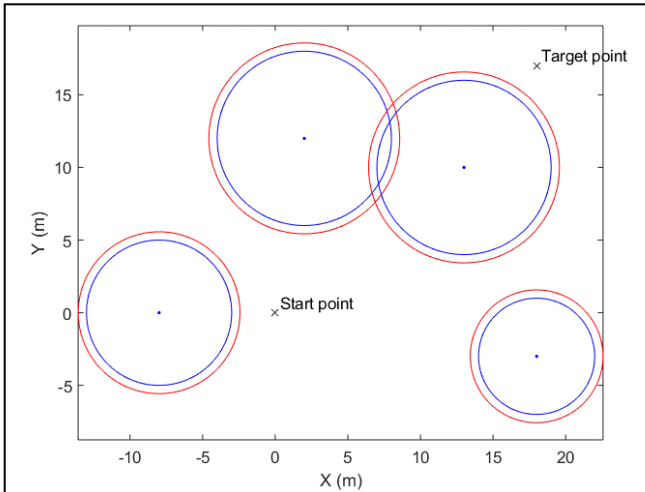


Fig 7 One by one obstacle representation in metres: start point S (0, 0); target point T (18, 17); obstacle array XO = [-8, 2, 18, 13], YO = [0, 12, -3, 10]; obstacle radius array RO = [5, 6, 4, 6]; determinant radius array RD = [5.571, 6.571, 4.571, 6.571].

produces a shorter but more complicated path with eleven line-segments. The projection method generates a path 4.63 m longer but with only three line-segments. The computational effort is noticeably reduced by merging the five obstacles into one – over six times faster.

3) Dividing an obstacle

For a large concave obstacle, using a single large radius circle to represent is far from reality. Instead, one obstacle can be modelled by two ‘connected’ overlapping smaller circles using the projection method (Table IIIa, b). This representation reduces the length of the route by 14.35 m without increasing the path complexity considerably.

The current code cannot handle more than two connected overlapping circles in the representation in cases where the reference ST line intersects only an obstacle in between the ends of a chain of obstacles (Table IIIc). A horizontal translation of the line ST to the left in this particular case is required (Table IIId).

B. Single obstacle avoidance: projection and tangent methods

The tangent method produces shorter routes by keeping the manoeuvre points close to the obstacle, but it consumes more time and computational resource comparing to the projection method in the case shown in Table IV. Turning angle affects how close the path follows the obstacle boundary, and it differs between the two methods. Avoidance of a single obstacle requires at least three turning manoeuvres: the one at the start point S with a maximum of 180 degrees, the other two at the two manoeuvre points each having a maximum of 90 degrees. Although the maximums are the same for both methods, in most cases the turning angle is smaller calculated by the tangent method. From the turning angles f1, f2 and f3 shown in Table IV, the tangent method produces smoother path, requiring less control effort.

The relative obstacle size affects the behaviour of the two methods. Different obstacle configurations can be examined by keeping the start and target positions and changing the obstacle determinant radius (Table V). The centre of the obstacle is at the middle of the ST line which is 50-m long, and the coordinates for the start S (10, 20), target T (40, 60), and obstacle O (25, 40). It is possible to evaluate the design of the trajectories using the manoeuvre points. It is expected that over or under shoot of control around these points would occur while the vehicle changes direction. Two 1-m-radius (the same as the length of the Pytheas robotic boat) circles are visually overlaid on the manoeuvre points for each method. The total route length calculated using each method and different determinant radii (RD) are illustrated in Table V. The increase of the route length as percentage relative to the length of the reference line ST is shown.

TABLE II. MERGING MULTIPLE OBSTACLES

Five obstacles	One obstacle merged from five
Start point S (-5, 10); target point T (60, 30) in m	
Obstacle position array: XO = [10, 17, 25, 33, 40] YO = [20, 4.5, 36, 4.5, 20] Obstacle radius array: RO = [9.43, 7, 8, 7, 9.43] Determinant radius array: RD = [10, 7.57, 8.57, 7.57, 10]	Obstacle position: O (25, 20) Obstacle radius: 24.429 Determinant radius: 25
Straight-line ST (LOS) length: 68.01	
Number of segments: 11 Route length: 97.31	Number. of segments: 3 Route length: 101.94
Computational time: 2.547 s	Computational time: 0.391 s

Once the size of the obstacle is small enough relative to the reference line ST, e.g. $RD = 5$ and 10 m in Table V, the tangent method produces a path that intersects the 1-m-radius safety circle for manoeuvre. By increasing the radius of the obstacle region by 1 m, the tangent method can retain its advantage for some cases (i.e. $RD = 10$ and 15 m in Table V) but can lose it in others (i.e. $RE = 5$ m in Table V) – see Table VI. The projection method suffers the similar problem in some obstacle layouts. For this reason, the safety radius addition is introduced. Nevertheless, the short route produced by the tangent method could not outweigh the higher risk of manoeuvre points being very close to the obstacle boundary.

TABLE III. DIVIDING AN OBSTACLE INTO TWO OR MORE

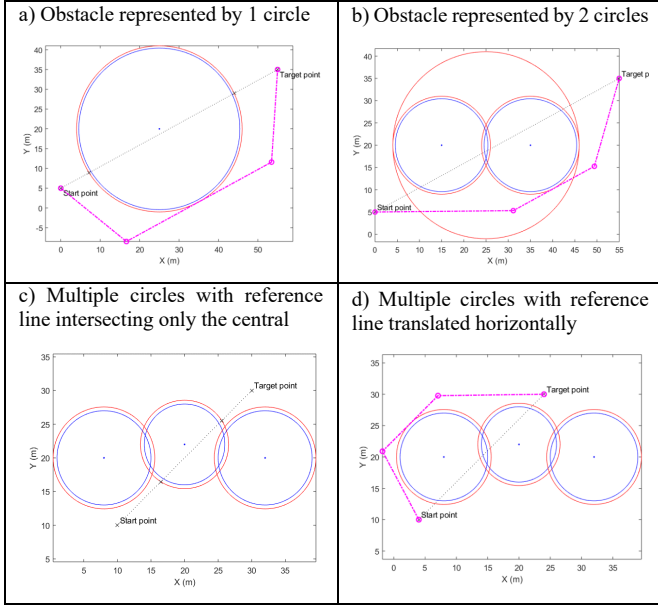


TABLE IV. PROJECTION AND TANGENT METHODS

Projection method	Tangent method
Start Point: S (15, 30) Target Point: T (30, 45) Obstacle position: O (23, 36) Obstacle radius: 4.429 m Determinant radius: 5 m Reference line ST length: 21.21 m	
Route length: 23.27 m	Route length: 22.50 m
Extra distance travelled: 2.06 m	Extra distance travelled: 1.29 m
$f1 = f2 = 0.613$ rad, $f3 = 0.503$ rad	$f1 = f2 = 0.382$ rad, $f3 = 0.330$ rad
Computational time: 0.2310 s	Computational time: 0.2116 s

C. Multiple obstacle avoidance: segment method

The parallel method can produce a shorter route than that calculated from the segment method when the order to avoid obstacles is apparent and predefined. However, the segment method does not require this cumbersome preparation. The segment method is able to select a shorter route if all permutations are considered, but at the cost of computational time in comparison to the parallel method. On the other hand, a critical weakness of the current parallel method has been its focus on going around each and individual obstacles rather than overlooking the overall obstacle layout which is the advantage of the segment method. Table VII illustrates this with four obstacles aligned with increasing size where the parallel method produces 33 % longer route comparing with segment method. The computational time 6.6 s of the segment method is 5 times longer but still adequate for a vehicle with a speed less than 3 m/s. The focus of the multiple obstacle avoidance would be to improve the segment method.

The segment method computes solution based on all possible orders of the obstacle array so as to ensure the shortest route. For 4 obstacles, there 24 possible orders. Some of the candidate solutions are provided in Table VIII.

The segment procedure of trying all permutations has been thorough, but the algorithm cannot converge at certain obstacle configurations such as the one illustrated in Table IXa where no solution can be reached using the current segment method. Representing irregular obstacles with more than two circles is beneficial. A logic similar to [7] has been developed in the updated segment algorithm using virtual ‘intermediate’ waypoints (Table IXb). When the basic segment procedure is not able to reach a solution, an array with 6 virtual waypoints is created to overcome this limitation:

$$x_intermediate = [\min_x, \max_x, XS, XS, \min_x, \max_x]$$

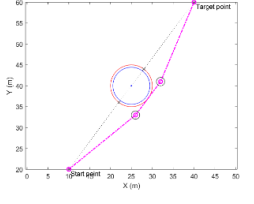
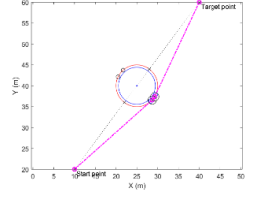
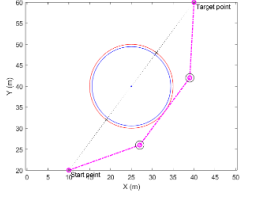
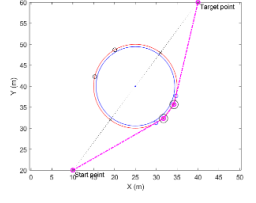
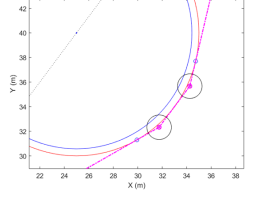
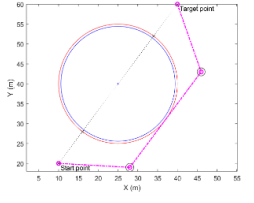
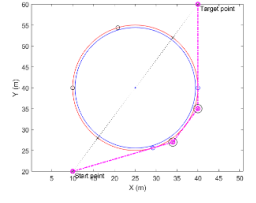
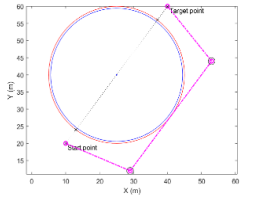
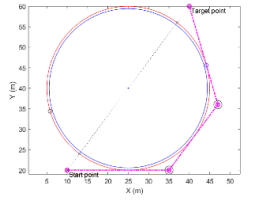
$$y_intermediate = [YS, YS, \min_y, \max_y, \min_y, \max_y]$$

where the minimum and maximum values correspond to the x and y coordinates, calculated by subtracting and adding the obstacle radii to the respective obstacle positions. YS and XS are the coordinates of the start point S.

In a more complicated obstacle layout shown in Table X, the basic segment method does not converge, and the updated procedure overcomes the limitation by examining the six intermediate virtual waypoints. The fifth waypoint (\min_x , \min_y) is found to produce a solution. In Table X, Part 1 of the path planning is designed from the start point to this intermediate virtual waypoint; Part 2 of the path is calculated starting from this intermediate virtual waypoint to the target point; Finally, the overall desired path is achieved.

Adding extra margin in the turning angle by 0.1 degree serves as a ‘padding’ to reduce the impact from the accumulated rounding error due to the use of tangent trigonometry functions. Without this treatment, whenever the segment algorithm perceives any small amount of intersection between the segment line and the obstacle, it will try to split the segment into smaller parts, giving rise to multitude of small segments increasing the complexity. Calculations adopting this approach show no apparent difference in the total route length, but considerably reduced segment number. The example in Table XI demonstrates a benefit of 14 less segments and only 0.008 m difference in total route length.

TABLE V. OBSTACLE SIZE (RD) AND I-M MANOEUVRE POINTS

RD (m)	Projection method (Route length percentage increase relative to ST)	Tangent method (Route length percentage increase relative to ST)
5	 (51.23 2.46%)	 (51.01 2.02%)
10	 (56.06 12.11%)	 Zoom-in:  (54.17 8.35%)
15	 (66.06 32.11%)	 (60 20%)
20	 (81.23 62.46%)	 (70 40%)

A path simplification and smoothing technique is incorporated to reduce the number of manoeuvre points after all routes are created by the updated segment method and the shortest route is chosen. The algorithm tries to connect two manoeuvre points by skipping any intermediate ones without intersecting any obstacles. If the segment created by connecting the n^{th} manoeuvre point with the $n+2^{\text{th}}$ does not intersect any obstacle, the $n+1^{\text{th}}$ is removed. This procedure is terminated when all manoeuvre points are examined. This procedure could be called after each candidate route is generated but the computation time would be increased considerably, so it is only deployed after the shortest route is identified. Table XII demonstrates the enhancement using a typical case that shows noticeable benefit with route length reduction by 23% and 67% less manoeuvre points.

TABLE VI. PROJECTION METHOD AND TANGENT WITH EXTRA MARGIN

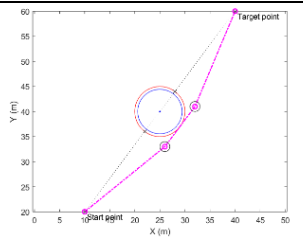
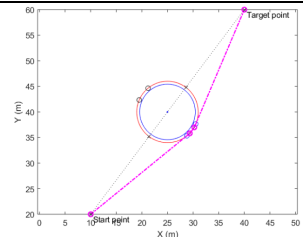
Projection method	Tangent method with extra margin in the safety radius
Start Point: S (10, 20); Target Point: T (40, 60)	
Obstacle position: O (25, 40)	
Obstacle radius: 4.43 m	Obstacle radius: 5.43 m
Determinant radius: 5 m	Determinant radius: 6 m
Straight-line ST length: 50 m	
Route length: 51.23 m	Route length: 51.46 m
Extra distance travelled: 1.23 m	Extra distance travelled: 1.46 m
	

TABLE VII. PARALLEL AND SEGMENT METHODS: SPECIAL CASE

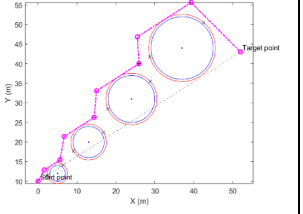
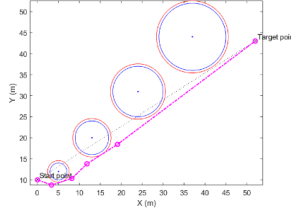
Parallel method	Segment method
Start Point: S (0, 10); Target Point: T (52, 43)	
Obstacle position array: XO = [5, 13, 24, 37], YO = [12, 20, 31, 44]	
Obstacle radius array: RO = [2, 4, 6, 8]	
Determinant radius array: RD = [2.571, 4.571, 6.571, 8.571]	
Reference line ST length: 61.59 m	
Number of line-segments: 9	Number of line-segments: 5
Route length: 84.111 m	Route length: 63.375 m
Computational time: 1.324 s	Computational time: 6.623 s
	

TABLE VIII. SEGMENT METHOD: CANDIDATE ROUTES

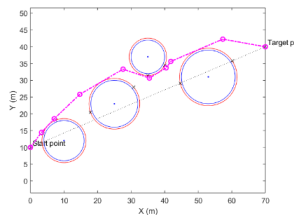
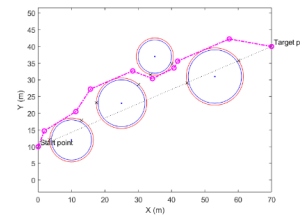
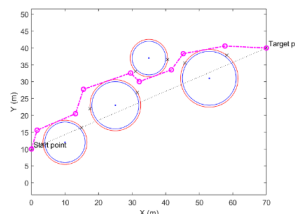
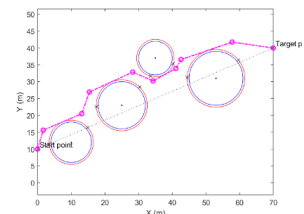
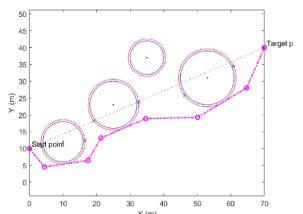
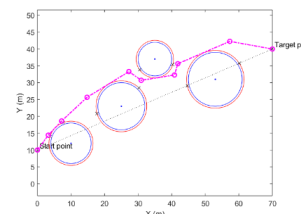
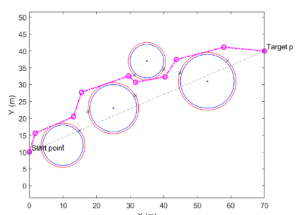
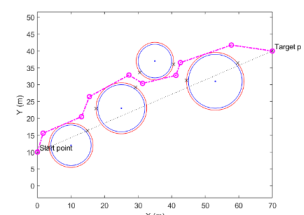
Route length (m) number of segments	
	
(82.81 9)	(83.61 9)
	
(85.55 9)	(85.00 9)
	
(88.02 7)	(83.98 9)
	
(85.38 9)	(85.51 9)

TABLE IX. SEGMENT METHOD: BASIC AND VIRTUAL WAYPOINTS

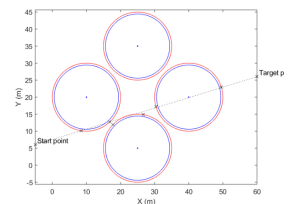
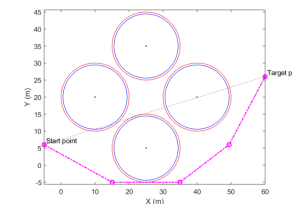
Segment method	Segment method with intermediate virtual waypoints
Start Point: S (-5, 6); Target Point: T (60, 26) m Obstacle position array: XO = [10, 25, 25, 40]; YO = [20, 5, 35, 20] Obstacle radius array: RO = [9.429, 9.429, 9.429, 9.429] Determinant radius array: RD = [10, 10, 10, 10] Straight-line ST length: 54.43	
No solution	Number of segments: 4 Route length: 83.611 m Virtual waypoint: (49.43, 6) Computational time: 13.438 s
a)	b)
	

TABLE X. SEGMENT METHOD WITH VIRTUAL WAYPOINTS

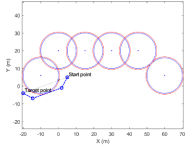
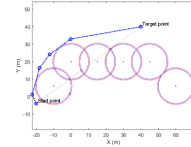
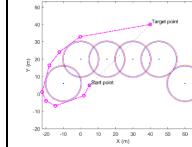
Part 1:	Part 2:	Final:
ST length: 26.57 m	ST length: 74.40 m	ST length: 49.50 m
Segments: 3	Segments: 5	Segments: 8
Route length: 30.34 m	Route length: 86.93 m	Route length: 117.3 m
		

TABLE XI. SEGMENT METHOD: TURNING ANGLE MARGIN

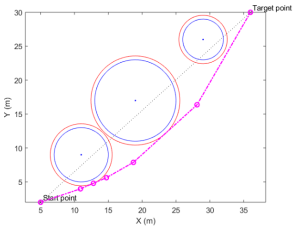
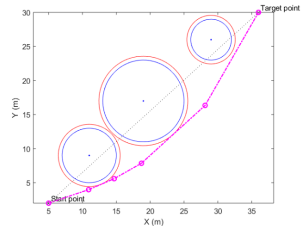
Segment method	Segment method with 0.1-degree turning margin
Start Point: S (5, 2); Target Point: T (36, 30) m Obstacle position arrays: XO = [11, 19, 29]; YO = [9, 17, 26] m Obstacle radius array: RO = [4, 6, 3] m Determinant radius array: RD = [4.57, 6.57, 3.57] m Straight-line ST length: 41.77 m	
Number of segments: 19 Route length: 43.386 m	Number of segments: 5 Route length: 43.394 m
a)	b)
	

TABLE XII. SEGMENT METHOD: PATH SIMPLIFICATION

Segment method	Segment method with path simplification
Start Point: S (5, 5); Target Point: T (40, 40) m Obstacle position arrays: XO = [-10, 0, 10, 20, 30]; YO = [6, 20, 20, 20, 6] m Obstacle radius array: RO = [10, 6, 6, 6, 10] m Determinant radius array: RD = [10.57, 6.57, 6.57, 6.57, 10.57] m Straight-line ST length: 49.50 m	
Number of segments: 9 Route length: 108.27 m Computational time: 8.224 s	Number of segments: 3 Route length: 83.75 m Computational time: 8.719 s
a)	b)

IV. DISCUSSION AND CONCLUSIONS

The present study provides a fundamental but necessary review and implementation of collision avoidance and path planning techniques base on 2D plane kinematics of vessels. The developed path planning logics are based on existing theoretical derivation from bottom up to individual solutions. Although control strategies and vehicle kinetics are not examined in the frame of work, a set of hydrodynamic parameters derived from computational fluidic simulations are provided for future development combining control dynamics into the path planning task. This section summarises the three objectives of obstacle representation, single obstacle avoidance, and multiple obstacle avoidance.

Using different sizes and cluster of circles seems to be the simplest form to represent environmental obstacles, but it proves to be complex due to its limitations imposed on the actual path planning algorithms. Use of the safety radius to take into account the dimensional, kinematic and dynamic constraints proves possible in most cases. Throughout the analysis only the dimensions of the Pytheas boat is included in the safety radius. The developed code allow adjustment to this radius for different obstacles which will benefit when a controller is considered in the simulation. By developing the single and multiple obstacle avoidance algorithms based on overlapping circular obstacle regions helped to minimize the area treated as non-navigable. Further enhancements of the environment could employ the Voronoi diagram process [8], but this relies on a reliable environmental dataset.

For single obstacle avoidance, both the projection and the tangent methods use three-line segments to guide the vehicle round a circular obstacle with the aim to minimise route length. The projection method produced a path with more room between manoeuvre points and the obstacle – a safer feature; the tangent method offers a path that is shorter, smoother and closer to the obstacle. The projection algorithm is preferred as it requires simpler code with reduced

computational effort – an important feature to scale up the algorithm for more complicated environment.

For multiple obstacle avoidance, both the parallel and the segment methods are derived from the projection method and they iteratively check the generated route does not intersect with any obstacles, with varied efficiencies. The parallel algorithm offers low computational complexity, but it is limited to certain obstacle layout configurations. This critical flaw shifts our development to the segment method. The segment algorithm is able to handle multiple obstacles without simplifying the environment or designing a path drifting far away from the actual positions of the obstacles. In the present study path planning is evaluated using the total route length, number of segments required, and computational time. The latter is sully comprised when the first two aspects of the evaluation is at more critical position. The intermediate virtual waypoint is added to the segment method to help the algorithm to realise the global pattern of the obstacle cluster. A further simplification process is also incorporated in the segment method to remove unnecessary manoeuvre points and therefore reducing route length and improving the smoothness of the route.

REFERENCES

- [1] Y Huang, Z Ji (2017). Autonomous boat dynamics: how far away is simulation from the high sea? IEEE Proceedings of OCEANS, Aberdeen, p2106-2113.
- [2] Z Liu, Y Zhang, X Yu, C Yuan (2016). Unmanned surface vehicles: An overview of developments and challenges. Annual Reviews in Control 41, p71-93.
- [3] C Goerzen, Z Kong, B Mettler (2010). A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance. Intelligent & Robotic Systems, p65-100.
- [4] D Ferguson, M Likhachev, A Stentz (2005). A Guide to Heuristic-based Path Planning. In: Proceedings of ICAPS Workshop on Planning under Uncertainty for Autonomous Systems, AAAI.
- [5] S Campbell, W Naeem, GW Irwin (2012). A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance manoeuvres. Annual Reviews in Control 36, p267-283.
- [6] ITTC Recommended Procedures and Guidelines (2011). Practical Guidelines for Ship CFD Applications 7.5-03-02-03.
- [7] J Sheng, G He, W Guo, J Li (2010). An Improved Artificial Potential Field Algorithm for Virtual Human Path Planning. Digital Techniques and Systems, p592-601. Berlin, Heidelberg: Springer.
- [8] H Niu, Y Lu, A Savvaris, A Tsourdos (2016). Efficient path planning algorithms for unmanned surface vehicle. IFAC-Papers OnLine 121-126. 10.1016/j.ifacol.2016.10.331